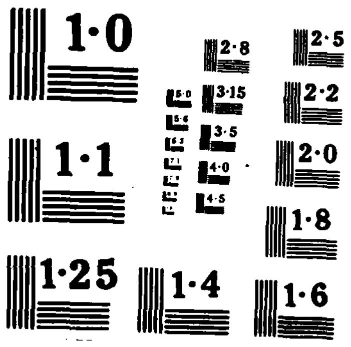
AD-A170 291 VALUE CELL ENCODING STRATEGIES(U) ROCHESTER UNIV NY DEPT OF COMPUTER SCIENCE J SULLINS AUG 85 TR-165 N00014-82-K-0193 1/1

UNCLASSIFIED F/G 9/2 NL


1·0
2·8
2·5
3·15
2·2
3·5
1·1
2·0
4·0
4·5
1·8
1·25
1·4
1·6

(12)

AD-A170 291

Value Cell Encoding Strategies

John Sullins
Computer Science Department
The University of Rochester
Rochester, New York 14627

TR 165
August 1985

rochester
Department of Computer Science
University of Rochester
Rochester, New York 14627

86 7 29 048

- A -

Value Cell Encoding Strategies

John Sullins
Computer Science Department
The University of Rochester
Rochester, New York 14627

TR 165
August 1985

Abstract Sub k-1

In many application areas, particularly in the biological sciences, there is the need to store several values of variables. Given a finite precison, one can store these values in N^k explicit cells, refered to as *value cells*, in a k-dimensional space of grain N. Typically, the number of values that must be stored is a very small fraction of the total number specified by the grain of the multidimensional space. This leads to data structuring that reduces the number of explicit cells required for a given level of accuracy. One idea is *coarse coding*, intersection of larger, coarser grained cells. Coarse coding has been shown to reduce the number of cells required by a factor of $1/D^{k-1}$ where D is the diameter of the coarse cell in units of fine grained cells. This intuitively appealing idea in fact involves many subtle tradeoffs that are the focus of this paper. Coarse coding is shown to be independant of the isptrophy of the cells and superior to simply reducing the grain of the representation space. Loss of information due to the possibility of some fine grained cells sharing some of the same coarse grained cells and due to uncertainty in the input and translation of data is examined. Sub k

This work was supportted in part by the Defense Advanced Research Projects Agency under grant number N00014-82-K-0193 and in part by the National Science Foundation grant number RCD-8450125.

1. Introduction

The encoding and access of values in a multidimensional space is an integral part of many computation systems, both mechanical and biological. For example, there are many inputs available to the visual system, such as position in space, angle of rotation, size, color, depth, velocity, etc. for a large number of points on the retina, that a system must be able to access frequently. These points must be stored with some great degree of accuracy, as well (for example, the eye is sensitive to positional changes of 5 arc seconds). Thus a central problem is: how can one represent a discrete set of m points in a euclidean space of dimension k?

One way of encoding a set of values is simply to list each value explicitly, in the form

$$
\begin{array}{ll}
((\text{coordinate}_1, \dots, \text{coordinate}_k), & \leftarrow \text{point } 1 \\
(\text{coordinate}_1, \dots, \text{coordinate}_k), & \leftarrow \text{point } 2 \\
\bullet & \\
\bullet & \\
\bullet & \\
(\text{coordinate}_1, \dots, \text{coordinate}_k)) & \leftarrow \text{point } m
\end{array}
$$

Unfortunately, accessing these points requires time O (km), which would make real-time implementation difficult. An alternative is to represent each value as a cell in a *discrete k-dimensional space.* Values can be accessed in O (1) time by a computer with parallel architecture or a biological connectionist network (Feldman and Ballard[1]). This strategy also has drawbacks, however, the most important being:

(1) The finite number of discrete values (the *grain*), places limits on the accuracy with which values can be represented. This has important consequences for biological computation systems in particular, for if there is a limit on how many different values can be encoded, then that limit applies to the amount of processes that can be run in parallel in the brain (which consists of value cells called neurons). To compensate for this, a very fine grain can be used.

(2) This strategy requires N^k discrete units for representation (where N is the grain). This problem is the more important of the two, as both mechanical and biological systems have limits on the amount of storage available. For example, 100 discrete values over each of 10 parameters is probably less than what the human brain uses for vision, but even this would require 10^{20} units of storage, which is far greater than the number of neurons in the human brain.

1.1. Overlapping Cells

One method of reducing the amount of cells involves *overlapping cells.* Overlapping receptive fields are a constant phenomenum in biology but only recently has work been done analyzing the usefulness of this representation for other applications. General consequences of computing with this kind of method are explored in Ballard[2]; the first

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER TR 165	2. GOVT ACCESSION NO. AD-A170 291	3. RECIPIENT'S CATALOG NUMBER
4. TITLE *(and Subtitle)* Value Cell Encoding Strategies		5. TYPE OF REPORT & PERIOD COVERED Technical Report
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) John Sullins		8. CONTRACT OR GRANT NUMBER(s) N00014-82-K-0193
9. PERFORMING ORGANIZATION NAME AND ADDRESS Computer Science Department The University of Rochester Rochester, New York 14627		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS DARPA/1400 Wilson Blvd. Arlington, VA 22209		12. REPORT DATE August 1985
		13. NUMBER OF PAGES 22 pages
14. MONITORING AGENCY NAME & ADDRESS *(if different from Controlling Office)* Office of Naval Research Information Systems Arlington, VA 22217		15. SECURITY CLASS. *(of this report)* Unclassified
		15a. DECLASSIFICATION DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT *(of this Report)*

Distribution of this document is unlimited

17. DISTRIBUTION STATEMENT *(of the abstract entered in Block 20, if different from Report)*

18. SUPPLEMENTARY NOTES

None

19. KEY WORDS *(Continue on reverse side if necessary and identify by block number)*

Hough Transform, parallel algorithms, distributed representations, connectionism.

20. ABSTRACT *(Continue on reverse side if necessary and identify by block number)*

In many application areas, particularly in the biological sciences, there is the need to store several values of variables. Given a finite precision, one can store these values in N^k explicit cells, referred to as value cells, in a k-dimensional space of grain N. Typically, the number of values that must be stored is a very small fraction of the total number specified by the grain of the multidimensional space. This leads to data structuring that reduces the number of explicit cells required for a given level of accuracy.

20. ABSTRACT (Continued)

One idea is coarse coding, intersection of larger, coarser grained cells. Coarse coding has been shown to reduce the number of cells required by a factor of $1/D^{k-1}$ where D is the diameter of the coarse cell in units of fine-grained cells. This intuitively appealing idea in fact involves many subtle tradeoffs that are the focus of this paper. Coarse coding is shown to be independent of the isotrophy of the cells and superior to simply reducing the grain of the representation space. Loss of information due to the possibility of some fine-grained cells sharing some of the same coarse-grained cells and due to uncertainty in the input and translation of data is examined.

analysis of their efficiency is given by Hinton[3].

This method decomposes a space of size N^k into D coarser grained spaces of size $(N/D)^k$ each by having each unit in the coarse spaces represent a k-dimensional cube D units on a side in the N^k fine space. The divisions in each of the coarse spaces are chosen in such a way that each point in fine spaces has a unique representation in the set of points that it corresponds to in the set of D coarse spaces. This is done by offsetting each of the D spaces by one diagonal unit. That is, the d/*[th/*] coarse array is offset an amount (d-1, ... ,d-1). Then, given the fine grained value, the D coarse grained cells can be determined by rounding with appropriate offsets:

$$\begin{aligned}
(x_0, y_0) &= x/D,\ y/D \\
(x_1, y_1) &= (x+1)/D,\ (y+1)/D \\
&\bullet \\
&\bullet \\
&\bullet \\
(x_{D-1}, y_{D-1}) &= (x+D-1)/D,\ (y+D-1)/D
\end{aligned}$$

Figure 1
Recovery of data in coarse coding (for k = 2 and D = 3)

The number of units required for this representation is N^k/D^{k-1}.

In principle the points in fine space need only be represented implicitly by the coarse units. That is, the coarse arrays *behave* as if fine points were represented. Points in the fine space may be recovered by allowing the points in each of the coarse spaces to vote into each of the D points in the fine space that it represents. The votes from the various coarse spaces will overlap and the original point (and that point only) will recieve D votes. Other points will receive less than D votes. This method of specifying the fine value we term the *intersection method.*

Because of the $1/D^{k-1}$ space savings, the idea of overlapping cells is attractive, but there are a number of interesting questions.

1) Uniform overlapping fields, where proportionate cell sizes are all similar (isotropic scaling) is one possible representation of multi-dimensional space, but there is also the possibility of subspaces (anisotrpic scaling). These alternatives are examined (in section 3), and are shown to be similar to isotropic scaling.

2) If the degree of overlap is too large false signallings can result. This is known as the *illusory conjunction* problem. Since the severity of this problem depends on the statistics of the signal, we explore the extent of this problem through simulations.

3) Finally, we distinguish between logical (or discrete) noise, which results from processes interfering by occupying too many points too near to each other in the space, and arithmetic (or continuous) noise, in which errors in the input and/or translation mechanisms cause votes for a single process to be spread over many

contiguous points instead of a single point in the space.

2. Overlapping Cells

2.1. Overlapping Cells and Noise

Overlapping fields are useful when there is a good deal of noise associated with the input, as noise can dramatically effect the usefulness of non-overlapping cells. Consider accumulating evidence for a single process where the individual measurements are subject to some probability of error. Problems will result if the noise variance is large compared to the cell size. Even if it were not, if the exact position of the process falls on the borderline between two cells, then some values will fall into one cell and others will fall into the other. In either case, the estimates for the correct position in the parameter space are scattered over nearby cells.

Figure 2
Noise in estimating cells

One could try to correct these problems by estimating the correct cell as the average of contiguous cells over a noise threshold T. This may not work, however, if the grain is too fine, as the data may be spread such that not all points are contiguous. This would cause the mechanism to interpret it as several processes, none of which having enough evidence to be important. Overlapping cells seem to solve these problems, as each cell sees more input. Since the evidence that a fine grained point receives from this representation is very roughly proportional to the inverse of the distance from points close to it (as those are more likely to fall into more cells that also represent that particular point), overlapping cells do a good job of compensating for both the error spread and borderline problems. For more precise mathematical details, see Section 4.

2.1.1. Ambiguity in Overlapping Cells

There is, of course, some loss of information associated with any reduction in the size of the representation. In this case, if enough of the overlapping cells that represent a point in the fine grained space have been activated by other points, then there is no way of knowing whether that point should be on or off.

Figure 3
Example of ambiguity in coarse coding
points 1 and 2 activate all overlapping cells of point 3

This kind of error could result from noise, but is more likely to be the result of two processes being too close together and sharing many of the same overlapping cells. The mechanism would not be able to seperate the two processes with any degree of accuracy.

Determining the probability of error is very difficult, as determining the probability that n irregular shapes do not overlap is an unsolved problem for greater than one dimension. Instead, a large number of simulations were run, the results of which are given in Table 1. The test runs involved N's of 4, 8, 16, 32, 64, and 128, k's of 2 - 8, and D's from $\sqrt{N}$ to N/2 by multiples of $\sqrt{2}$. Since determining whether error occured requires reconstructing the original N^k space (and the computer these runs were made on simply does not have that much memory), error was considered to have occured when two points shared any of the same overlapping cells. The numbers in the table represent the logarithm (base 2) of the minimum number of points needed to cause at least a 10% chance of error. This was obtained by making 50 runs and increasing the number of points by a factor of two until at least five of the runs reported error.

3. Subspaces

3.1. Definition

Another way to represent an N^k space is to split a k-dimensional space into its k component 1-dimensional spaces by projecting the data into its various parameters. This is, in effect, the ultimate anisotropic cell representaion as each cell has the size of the space in one dimension and size 1 in all of the rest. An example is shown in Figure 4, where points in an XY plane are projected onto X and Y axes. This *subspace* representation requires only k times the k^{th} root of the space required by the original representation, a substantial savings.

Figure 4
Two-dimensional subspace example

Again, it is impossible to reduce the size of a representation without some information loss. In this case, we no longer know which points in the subspaces correspond to each other. Thus, when reconstructing the original m points from this representation, we would have no choice but to associate *all* of the points in the k subspaces with each other. This would result in m^k points, or m^k - m *extraneous associations*, as shown in Figure 5. This is clearly unacceptable.

Figure 5
Ambiguity of multiple data in subspace

3.2. Coarse-fine connections

This problem of ambiguity could be solved by interconnecting all of the points in the two subspaces by intermediate units, but that would result in the original k-dimensional space. Instead, the solution is to connect large *areas* between the spaces. These *coarse connections* could be represented as an k-dimensional array, but with a much coarser grain than the subspaces that it connects. Points in this array would represent a connection between points that lie in the areas of the two subspaces (as partitioned by the coarse divisions). This new representation would require kN + $(N/D)^k$

units, where D^k is the size of the k-dimensional coarse connection array.

Figure 6
Recovery of data in coarse-fine subspace

Of course, ambiguity is still possible. If two connections occupy the same unit in the coarse connection space, there is still no way to tell which values of the parameters are associated with which for those values connected to that point, but the chances of this happening are greatly reduced. To determine the expected error, note that this is mathematically identical to the number of urns with two or more recing a number of balls in a number of urns at random and defining the number of errors to be the number of urns with two or more balls in them. The expected number of errors, then, given m points placed in $(N/D)^k$ discrete units in the coarse connection space, is

$$(D/N)^k({}_mC_2)(1 - (D/N)^k)^{m-2}$$

$$\text{where } {}_mC_n = m!/n!(m-n)!$$

as given in Johnson and Kotz[4].

3.3. Comparison with coarse coding

For examples of how many points this representation can reasonably store, in comparison to the results of overlapping coding, see Table 2 which contains results for the same parameters as in Table 1 computed with the above equation. At first glance the results look very similar. Coarse-fine coding is slightly more efficient, but only by a factor of four at the extreme of the range measured. This is to be expected, perhaps, as there will always be a certain amount of information lost when the size of the representation space is reduced, and the loss is probably far more dependant on that reduction than on the method of representation. If this is the case, then if we use coarse coding (for which no simple error model exists) we can estimate the expected number of errors using the coarse-fine coding model above.

4. Continuous Noise

4.1. Sources of Continuous Gaussian Noise

So far we have been looking at *discrete* noise, interference produced by discrete processes running in the same space. If the cells themselves represent discrete values (such as high level concepts like "apple" and "orange"), then we would expect one cell and only one cell to be activated by some input. If the cells represent continuous values such as the position of an object in space or the frequencies corresponding to its color, however, then we would expect a certain amount of uncertainty concerning which cells are to be activated corresponding to the uncertainty in measuring these quantities in the first place, and this *continuous* noise is additional to, and must be modeled differently from the discrete noise described above. For example, most images in the real visual world have a certain amount of "blurring" associated with them, where the measurement process can be modelled as an ideal value corrupted by noise. When these images are translated to a discrete encoding, they may activate cells close to but not exactly the correct cell for the process. In general, such random noise from the real world is gaussian, so if we coarsely encoded a gaussian distribution of votes over a set of fine cells (in this example the retina) we would expect the votes for the coarse cells to exhibit that distribution.

4.2. Roundoff Error

Another source of error occurs when translating from one ecoding scheme to another. In figure 5 we are translating space parameters (x, y) to line partameters (r, T). In this example, a cell in rT space may correspond to several cells in xy space, but since each cell in xy space represents a *region* instead of a single point it may not be possible to partition these regions in such a way that any two regions in xy space corresponding to two cells in xy space overlap exactly when translated into the same cell in r space. In a case like this a cell in one space may not correspond *exactly* to a cell in another. It may correspond mostly to one cell, but partly to others depending on how the regions are translated. To compensate for this we could connect a cell in one space to several in the same region of another space (instead of just one), and give votes to the cells in such a way that they make up a gaussian ditribution centered around the translation of the region of the cell in the first space. Thus, if a region of a cell in one space lies on the "edge" of the corresponding cell in the other space, we would give that edge more "weight" by giving votes to adjoining cells on that edge. This can be done by 1) computing a gaussian curve based at the exact point that the data translates to and 2) giving each cell a vote corresponding to the integral of that curve over the volume of the cell.

Figure 7
Continuous noise in translation

4.3. Setting Thresholds to Eliminate Continuous Noise

As mentioned earlier, the acuity, or overall noise resistance of the system may be greatly improved by coarse coding. Instead of having to rely on a number of votes exactly choosing the correct cell, we can now rely on the effects of "near misses". More precisely, we can give coarse cells a certain threshold τ, and consider a fine cell to be activated if all of the coarse cells that represent it have reached some given *activation threshold*, as opposed to setting a threshold for the individual fine cells themselves. In order to eliminate all noise, the threshold must be chosen large enough so that all coarse cells corresponding to the correct fine cell (which we are assuming is at the center of the gaussian noise distribution) are activated, but small enough so that none others are activated. Of these two goals the first is far more important, as it is far better for a process to have some noise associated with it that is is for the process to disapear altogether.

First, we will determine the lower bound necessary to activate the correct cell. Given a standard gaussian correct cell-centered noise model, the expected number of overall votes needed to activate all coarse cells needed will depend on the expected number of votes needed to activate the cell farthest from the center of the curve (the cell expected to recieve the least number of votes). In the worst possible case this will be a coarse cell in which the correct cell lies in one of the corners.

Figure 8
Worst case coarse cell arrangement for cell activation

Given a gaussian vote distribution defined by Ae^{-r^2/σ^2} we can define the expected number of votes recieved by a coarse cell to be the sum of the distribution over the area of the coarse cell. In the case of a "corner cell", this would be from -1/2 to D-1/2 in all k dimensions, or

$$\int\int\int_{-1/2}^{D-1/2}\ldots Ae^{-(x1^2+x2^2+\ldots+xk^2)/\sigma^2}\,dx1\,dx2\,\ldots\,dxk$$

which is

$$\tau_1 = A(2\sigma/\sqrt{\pi}(\mathrm{Erf}(1/2\sigma) + \mathrm{Erf}((2D-1)/2\sigma))^k$$

where Erf(x) is the error function, which has derivative $2e^{-x^2}/\sqrt{\pi}$.

The next step is to find an upper bound on the number of votes that a coarse cell *not* overlapping the correct fine cell recieves. If any of these cells are activeted then other fine cells besides the correct one will be indicated, so our threshold must be great enough so that cell is not activated. Unfortunately, the exact pattern of coarse cells that determine a fin^ ell is not fixed, as shown in figure 9:

Figure 9
Different ways that coarse cells can combine to specify fine cells

so the pattern of coarse that determined the worst case "corner cell" above may not be the same pattern that determines the worst case cell not overlapping the correct fine cell. That case occurs when the correct cell is adjecent to a coarse cell in one dimension and in the center of that cell in the other k-1 dimensions, as in figure 10.

Figure 10
Worst case for non-activated coarse cell

Since this "side cell" is actually closer to the center of the gaussian curve than the corner cell in most of the dimensions, it is clear that it will almost certainly recieve more votes.

This means that it is impossible to choose a threshold that activates all correct coarse cells and none of the incorrect ones. Since we must activate all correct coarse cells, we will have to accept some incorrect cells, and some uncertainty concerning which of the corresponding fine cells is the correct one. This can be compensated for by averaging the values of a local group of activated cells in order to estimate the correct value.

Thus it appears that the best way to correct this problem is to find a way to reduce σ, the size of the noise spread. One way that the human visual system does this is to use the time dimension to augment the k-dimensional space. Cells further away from a point in the space are sampled *less often* than those closer to the point when determining the number of votes for a point. This means that a cell would **not** recieve as many votes from an input that lies in one of its corners as it would from one that lies in its center over a period of time.

Another solution would be to augment these logical receptive fields with *arithmetic* receptive fields that would determine the average position of activation over some range of cells cell_i. This is done by calculating

$$\text{value} = \frac{\sum_i v_i \times \text{val}_i}{\sum_i v_i}$$

where v_i is the number of votes for cell_i and val_i is the numerical value that cell_i represents, for each cell_i with v_i greater than some fixed threshold T.

5. Vernier Representation

Another possible use of these strategies is as a *vernier technique* [5]. If one wanted to store a number of exact values (more exact than the grain will allow), and one wanted parallel access to the **approximate** value of that value, then the approximate value could be stored in a very coarsely grained space, with a connection from that cell to a sequential access representation of the exact value.

Figure 11
Vernier representation

6. Conclusions

Overlapping coding is a method of representing high-dimensional values in a relatively small space where values may be recovered in parallel time. As may be expected, reducing the size of the representation space causes a certain amount of information to be lost, and this loss is more or less indepentant of the isotrophy of the representation.

When choosing the coarseness of the grain (D) to use with these methods, the user must balance the costs of added space against the costs of error. If a relatively noise-resistant system such as the Hough transform is used, then very large D may be chosen and a considerable amount of space may be saved. (Such a system has been implemented for two-dimensional image recognition. In that case D was set to $\sqrt{N}$, thus saving space by a factor of $\sqrt{N}$, without causing any significant loss of accuracy.)

References

[1] J. A. Feldman and D. H. Ballard, "Connectionist Models and Their Properties," *Cognitive Science, 6,* pp. 205-254, 1982.

[2] D. H. Ballard, "Cortical Connections and Parallel Processing: Structure and Function", TR 133, 1985.

[3] G. E. Hinton, "Shape Representations in Parallel Systems", *Proc. 7th IJCAI,* Vancouver, B.C., pp. 1088-1096, 1981.

[4] N. L Johnson and S. Kotz, **Urn Models and Their Applications**, John Wiley & Sons, New York, 1977.

[5] D. H. Ballard, Personal Comunication.

Table 1

N = 4

Log (minimum number of points causing 10% error)							
	k =						
D =	2	3	4	5	6	7	8
2	1	1	2	2	3	3	3

N = 8

Log (minimum number of points causing 10% error)							
	k =						
D =	2	3	4	5	6	7	8
3	1	2	2	3	3	4	4
4	1	1	1	2	2	3	3

N = 16

Log (minimum number of points causing 10% error)							
	k =						
D =	2	3	4	5	6	7	8
4	2	2	3	4	5	6	7
6	1	2	2	2	3	4	5
8	1	1	1	1	2	2	2

N = 32

Log (minimum number of points causing 10% error)							
	k =						
D =	2	3	4	5	6	7	8
6	2	2	3	4	6	7	8
8	1	2	3	4	4	5	6
11	1	2	2	2	3	3	4
16	1	1	1	2	2	3	3

N = 64

Log (minimum number of points causing 10% error)							
	k =						
D =	2	3	4	5	6	7	8
8	2	3	4	5	7	8	9
11	2	3	4	4	6	7	8
16	1	2	3	3	4	5	6
23	1	1	2	2	3	3	4
32	1	1	1	1	2	2	2

N = 128

Log (minimum number of points causing 10% error)							
	k =						
D =	2	3	4	5	6	7	8
11	2	4	5	7	8	10	11
16	2	3	5	5	6	8	9
23	1	2	3	4	5	7	8
32	1	2	3	3	4	5	6
45	1	1	2	3	3	4	4
64	1	1	1	2	2	3	3

Table 2

N = 2

Log (minimum number of points causing 10% error)							
	k =						
D =	2	3	4	5	6	7	8
2	1	1	1	2	2	3	3

N = 8

Log (minimum number of points causing 10% error)							
	k =						
D =	2	3	4	5	6	7	8
3	1	2	3	4	4	4	5
4	1	1	1	2	2	3	3

N = 16

Log (minimum number of points causing 10% error)							
	k =						
D =	2	3	4	5	6	7	8
4	2	3	3	4	5	6	7
6	1	2	3	4	4	4	5
8	1	1	1	2	2	3	3

N = 32

Log (minimum number of points causing 10% error)							
	k =						
D =	2	3	4	5	6	7	8
6	2	3	5	6	7	8	10
8	2	3	3	4	5	6	7
11	1	2	3	4	4	4	5
16	1	1	1	2	2	3	3

N = 64

Log (minimum number of points causing 10% error)							
	k =						
D =	2	3	4	5	6	7	8
8	3	4	5	7	8	10	11
11	2	3	5	6	7	8	10
16	2	3	3	4	5	6	7
23	1	2	3	4	4	4	5
32	1	1	1	2	2	3	3

N = 128

Log (minimum number of points causing 10% error)							
	k =						
D =	2	3	4	5	6	7	8
11	3	5	6	8	10	11	13
16	3	4	5	7	8	10	11
23	2	3	5	6	7	8	10
32	2	3	3	4	5	6	7
45	1	2	3	4	4	4	5
64	1	1	1	2	2	3	3

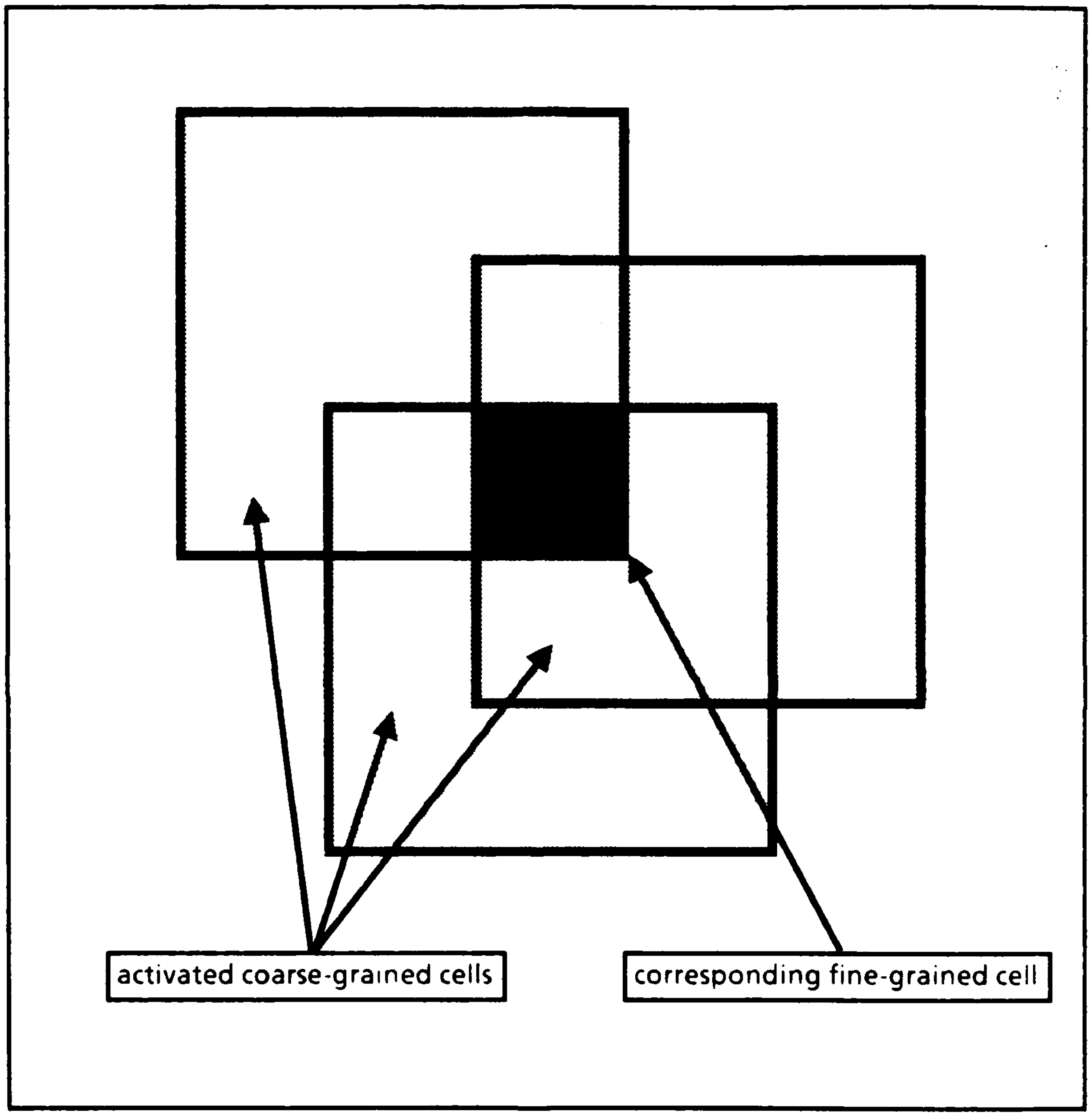

activated coarse-grained cells
corresponding fine-grained cell

Figure 1
Recovery of data in coarse coding (for k = 2 and D = 3)

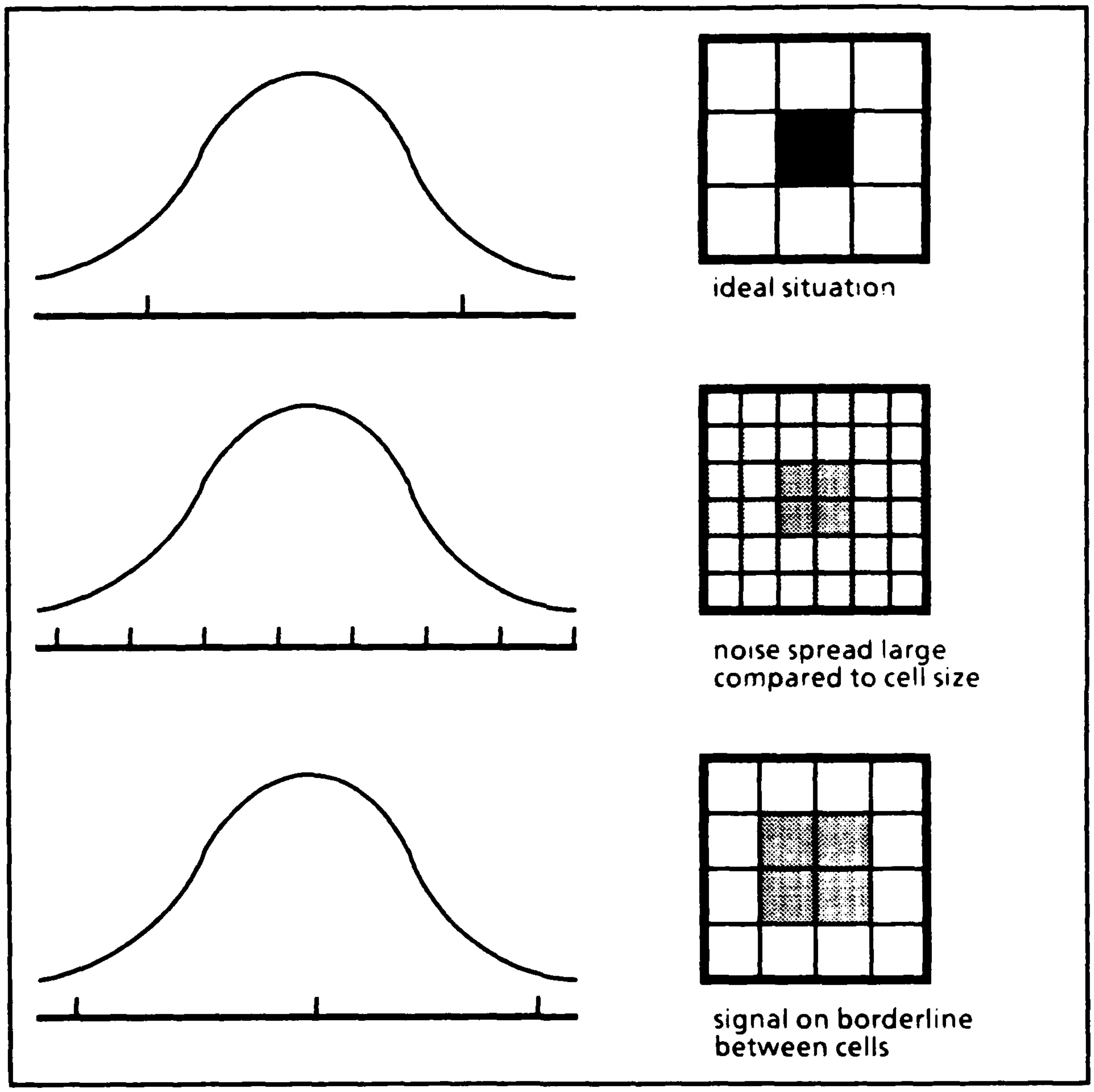

ideal situation
noise spread large compared to cell size
signal on borderline between cells

Figure 2
Noise in estimating cells

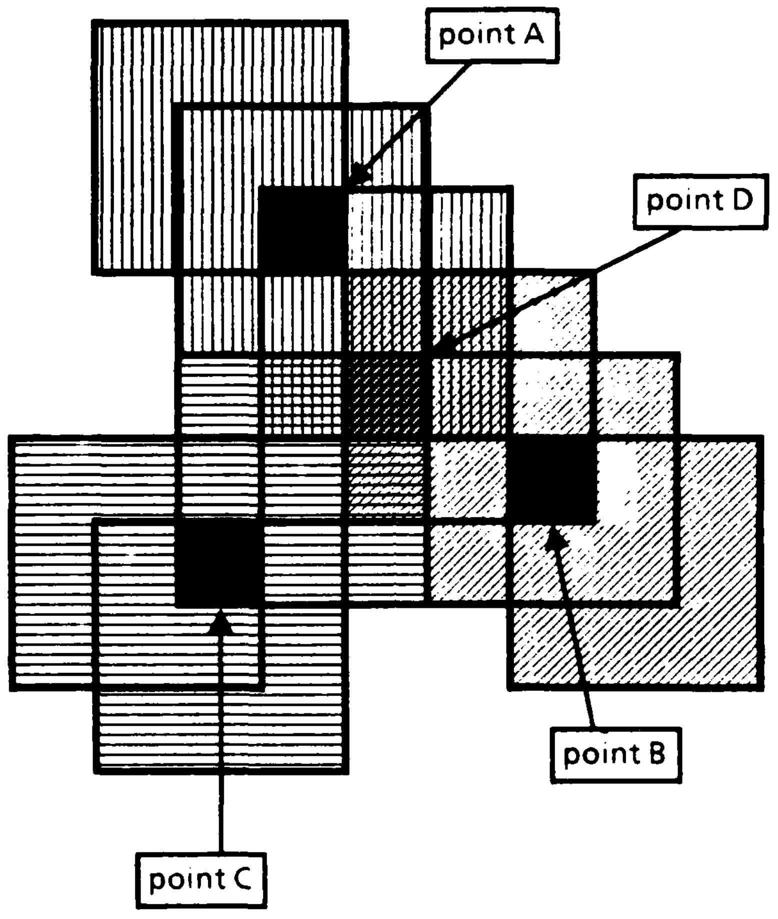


Figure 3
Example of ambiguity in coarse coding.

When points A, B, and C are activated, they activate all coarse cells necessary to activate point D. Thus, we cannot determine whether point D is actually activated.

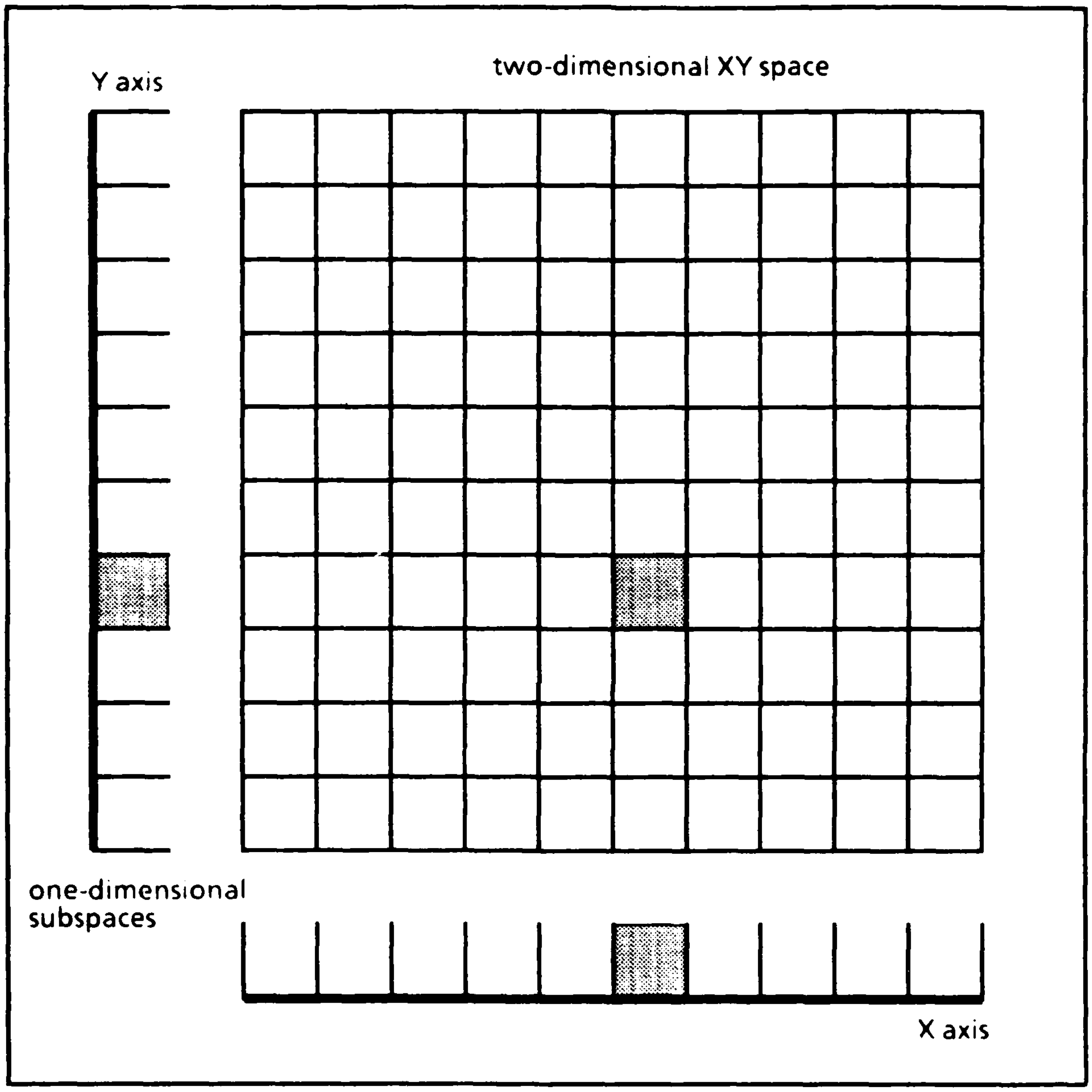


Figure 4
Two-dimensional subspace example

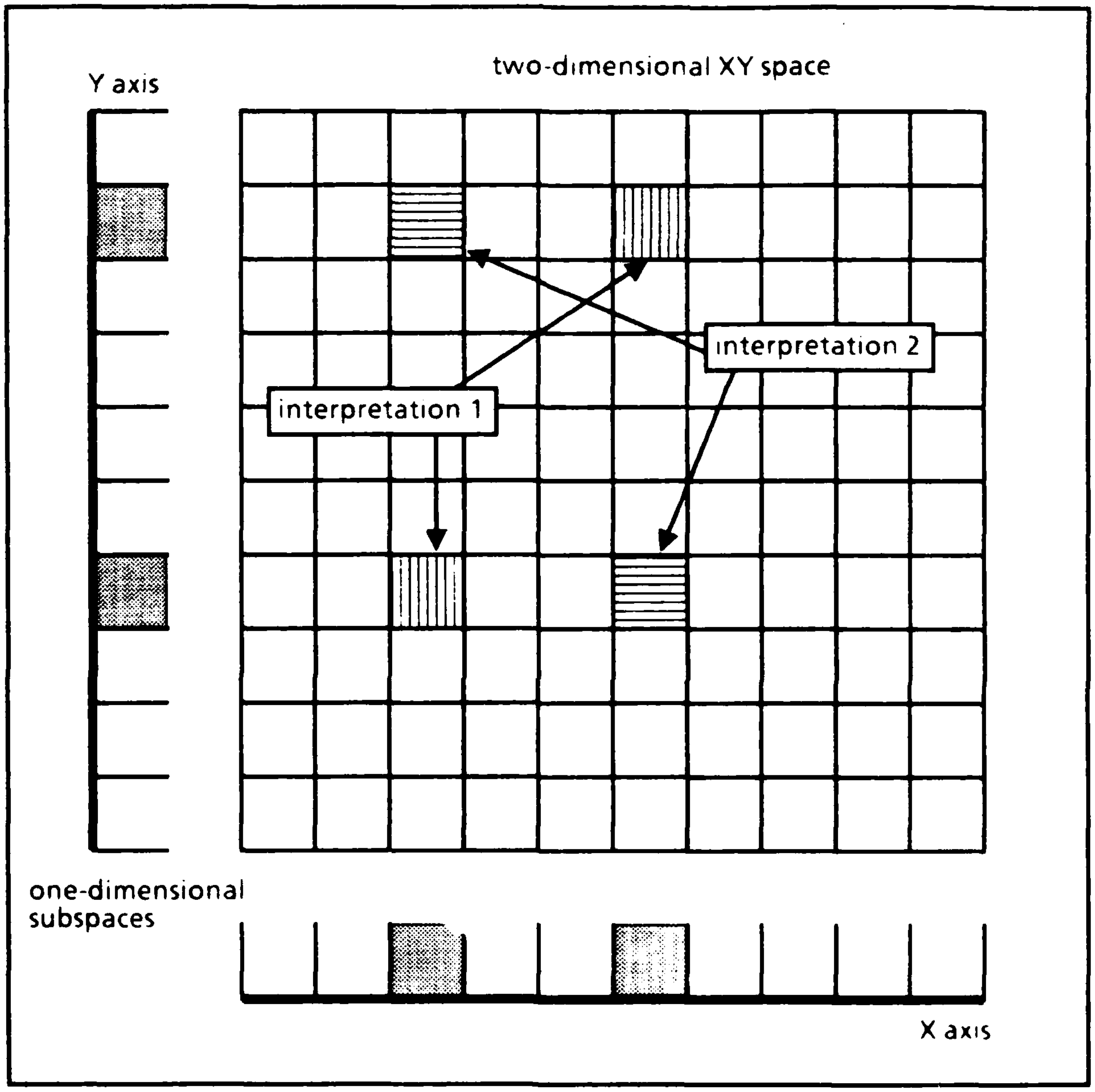


Figure 5
Ambiguity of multiple data in subspaces

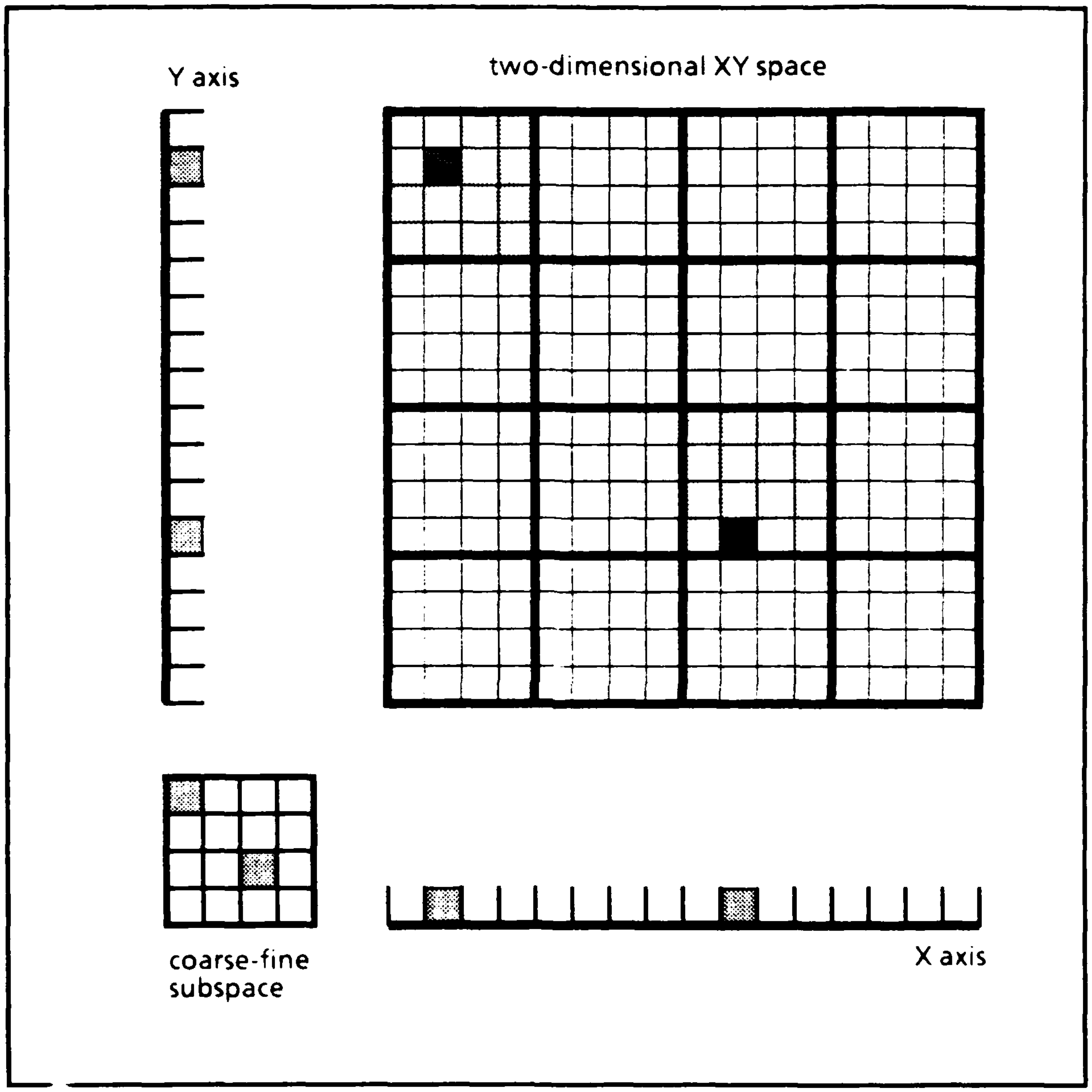


Figure 6
Recovery of data in coarse-fine subspace

Associations between points in X and Y subspaces
must lie within an activated region in the coarse-fine subspace.

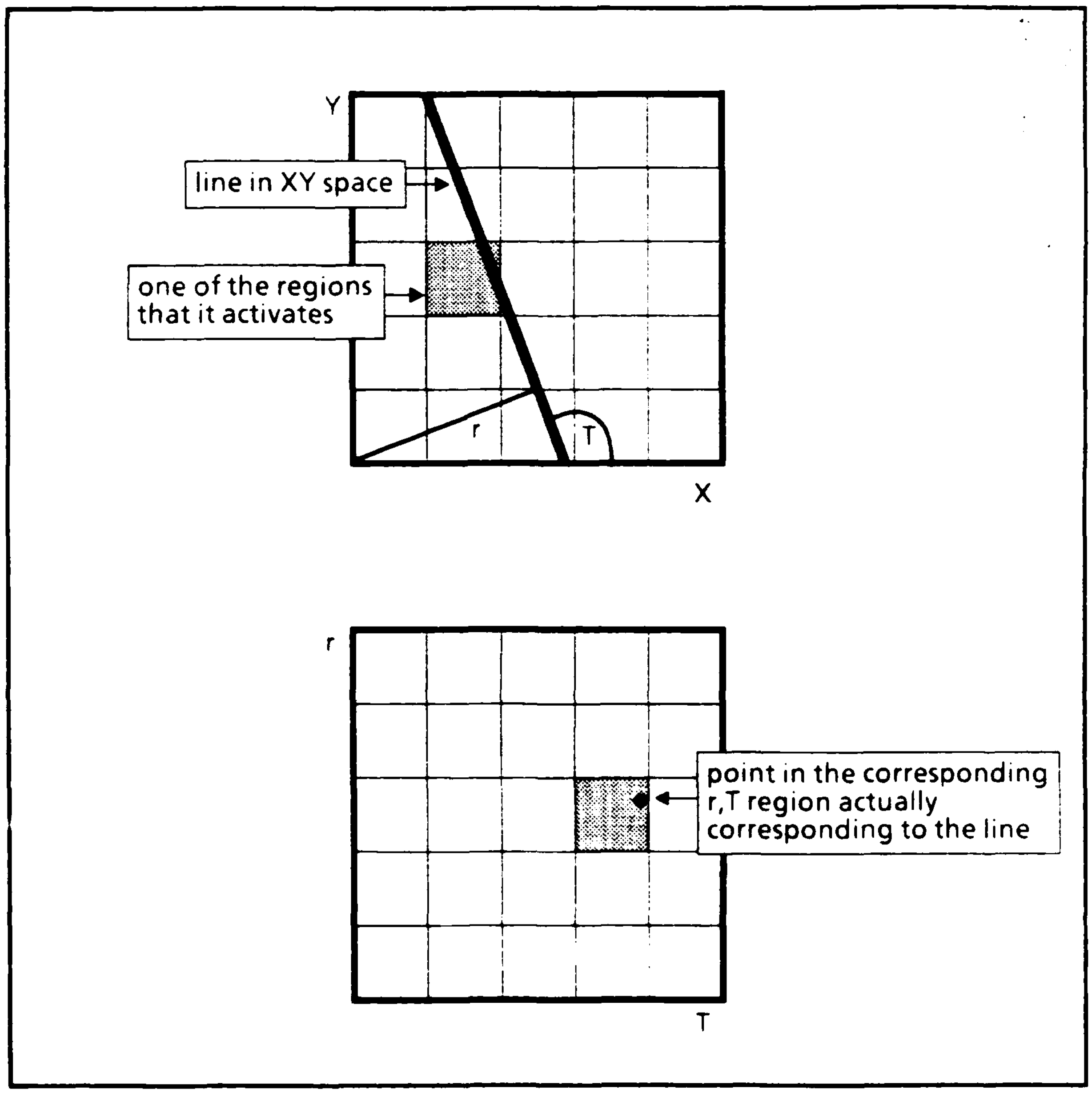

Y
line in XY space
one of the regions that it activates
r
T
X
r
point in the corresponding r,T region actually corresponding to the line
T

Figure 7
ontinuous noise in translation

An edge xyΘ might map into the peripherary of the r,T cell. Weights can compensate for this.

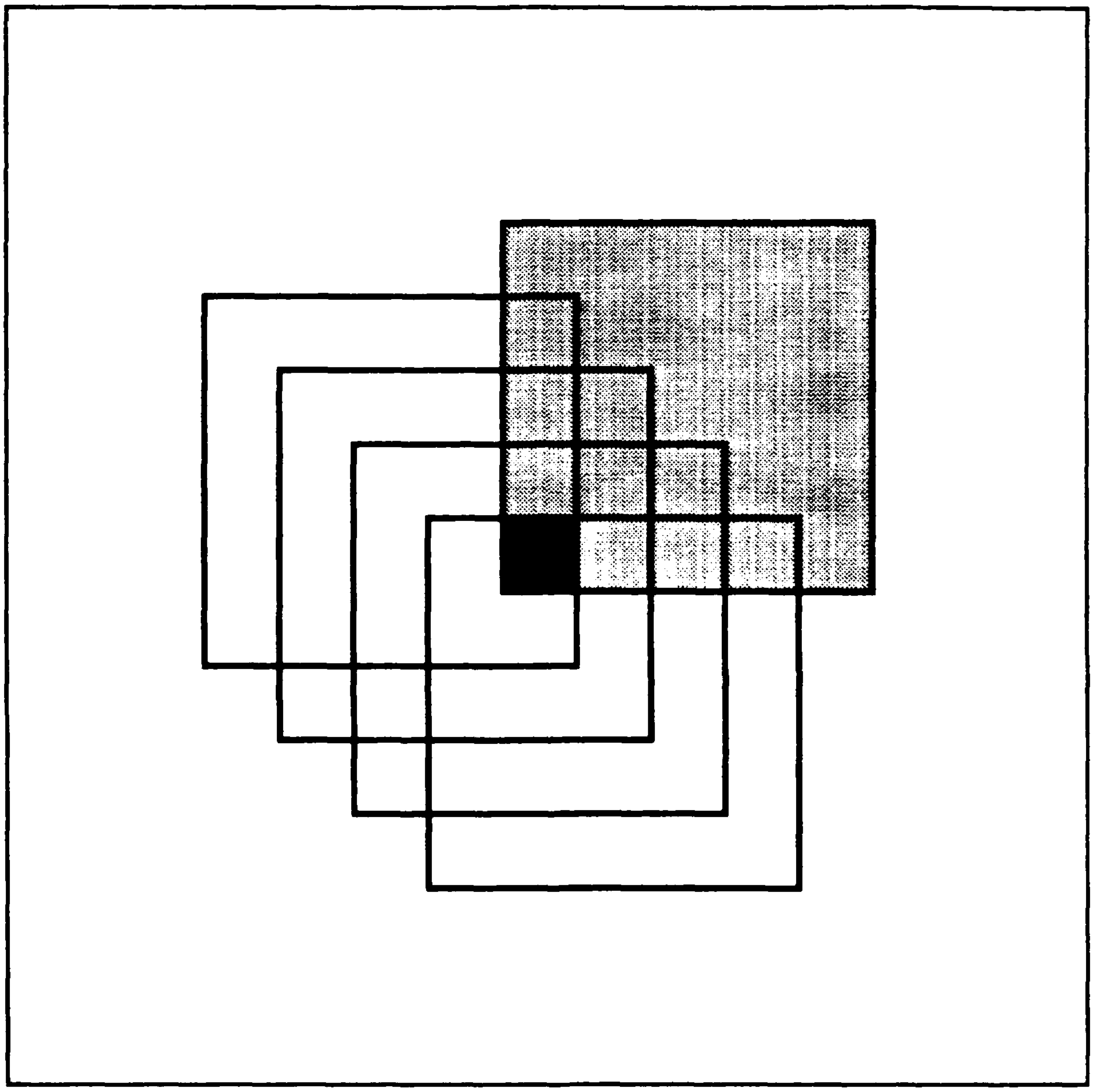


Figure 8
Worst case coarse cell arrangement for cell activation

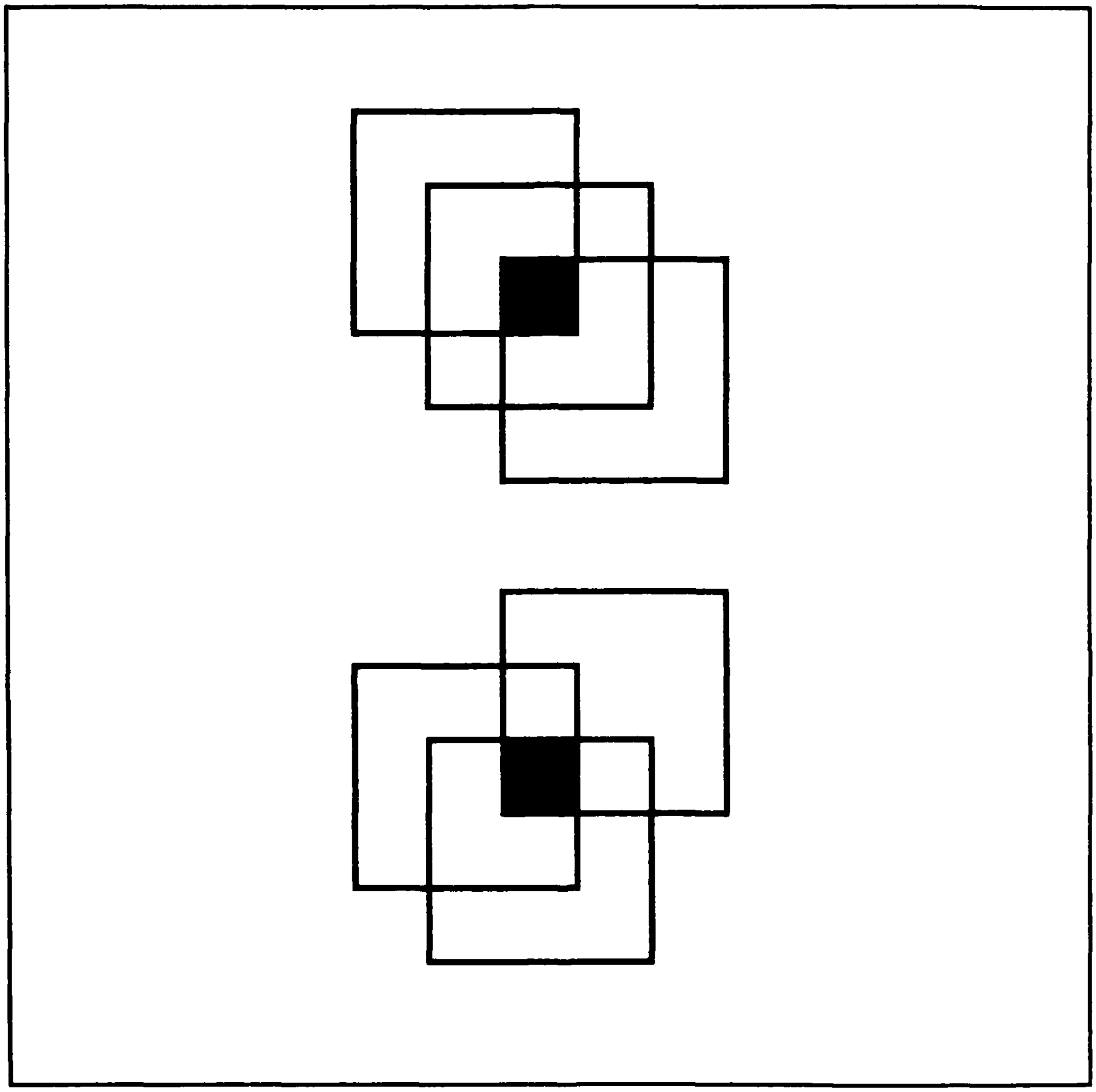


Figure 9
Different ways that coarse cells can combine to specify fine cells

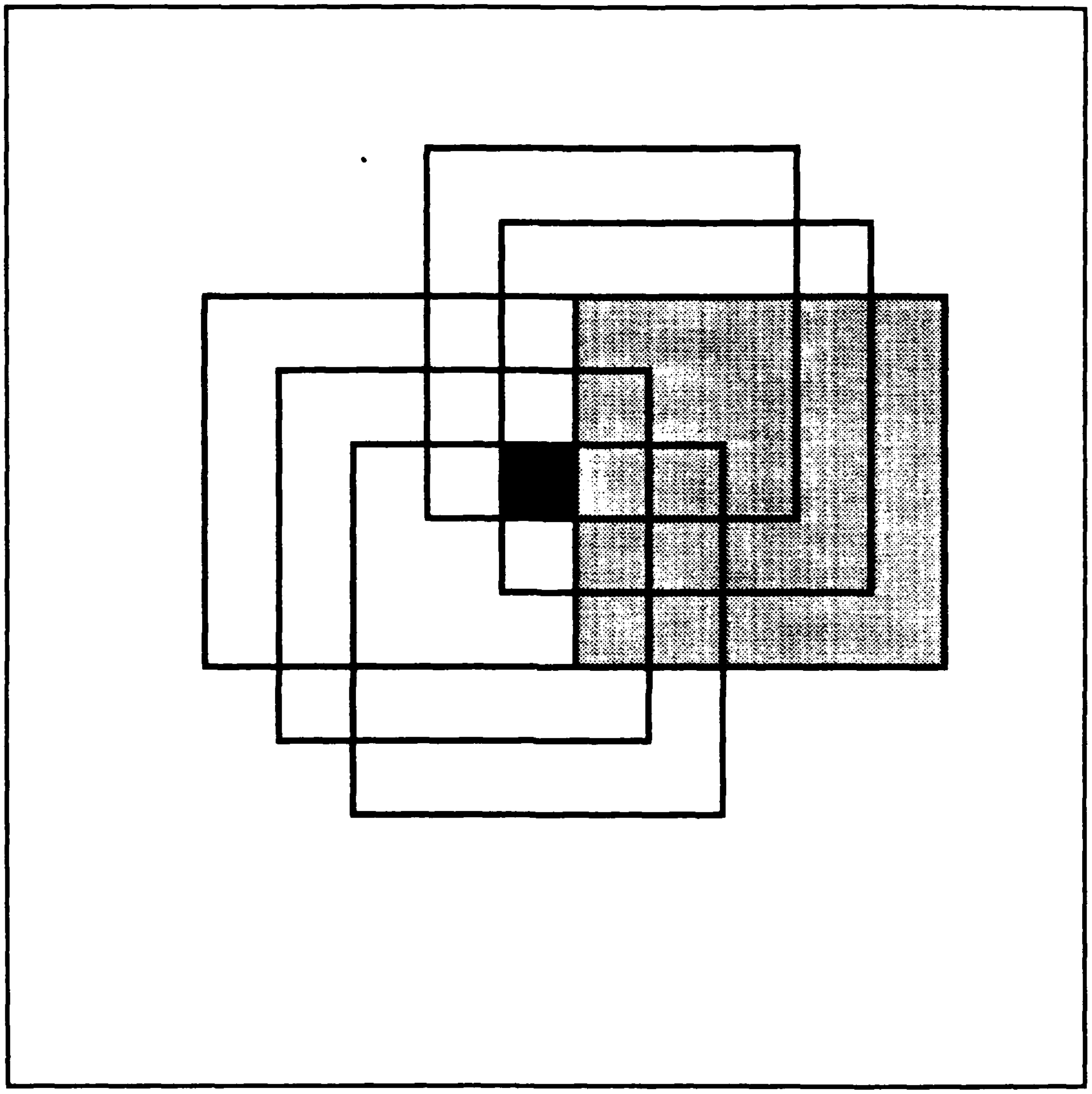


Figure 10
Worst case for non-activated coarse cell

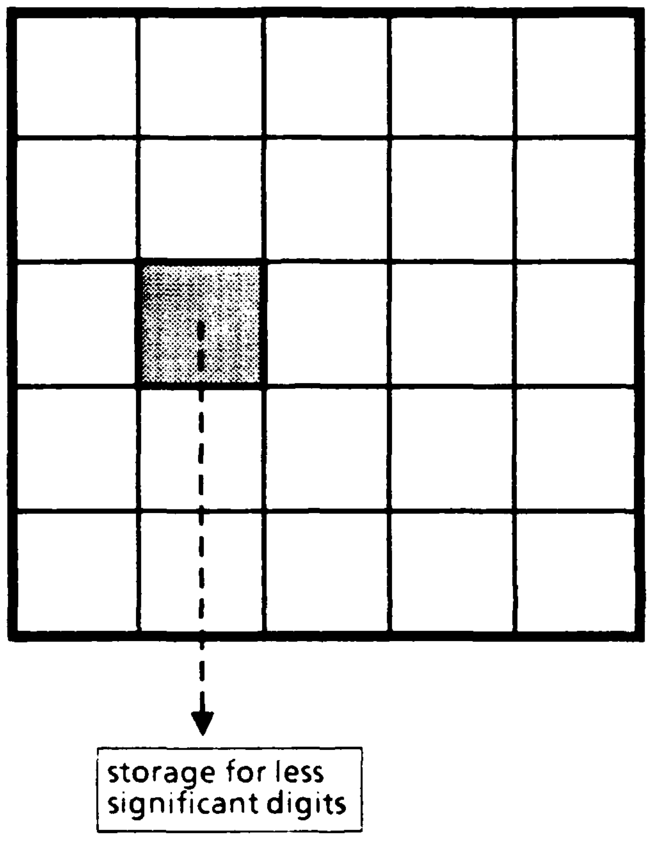


Figure 11
Vernier representation

END
DTIC
8-86